Руководство разработчика по

Приложению для работы с базой данных о достопримечательностях России

Разработчики:

Царегородцева Елизавета

Мотявин Матвей

Оглавление

[Технические требования 3](#_Toc42891869)

[Версии Библиотек 3](#_Toc42891870)

[Архитектура Приложения 4](#_Toc42891871)

[Структура Каталогов 5](#_Toc42891872)

[Листинг Скрипта 6](#_Toc42891873)

**Автор: Царегородцева Елизавета Редактор: Мотявин Матвей**

Технические требования

64-битная операционная система Windows, на которую возможна установка интерпретатора Python 3.7 (<https://www.python.org/downloads/>) .

Версии Библиотек

Для написания приложения были использованы следующие библиотеки Python (Табл. 1).

|  |  |
| --- | --- |
| Библиотека | Версия |
| sys | - |
| os | - |
| tk | 8.6.8 |
| numpy | 1.18.1 |
| matplotlib | 3.1.3 |
| requests | 2.23.0 |
| pandas | 1.0.3 |
| beautifulsoup4 | 4.8.2 |
| re | - |
| random | - |

Табл. 1. Версии библиотек

Архитектура Приложения

Данное приложение состоит из 7 модулей, которые находятся в каталоге Work/Scripts (Табл. 2). Главный модуль main.py содержит основной код программы. В этом же каталоге находятся специализированные скрипты с функциями, которые вызывает модуль main.py и другие модули.

|  |  |  |
| --- | --- | --- |
| Модуль | Местонахождение | Функция |
| main.py | Work/Scripts | Строит главное окно и вызывает функции из других модулей |
| get\_data.py | Work/Scripts | Парсинг сайта и сохранение базы данных в формат xlsx |
| first\_tab.py | Work/Scripts | Строит первую вкладку главного окна |
| second\_tab.py | Work/Scripts | Строит вторую вкладку главного окна |
| base\_statistics.py | Work/Scripts | Выводит и сохраняет базовую статистику |
| out\_pivot\_table.py | Work/Scripts | Строит сводную таблицу |
| group\_diagram.py | Work/Scripts | Выводит и сохраняет столбчатую диаграмму |

Табл. . Модули приложения

Структура Каталогов

Данное приложение использует следующую систему каталогов (Табл. 3).

|  |  |  |
| --- | --- | --- |
| Первый уровень | Второй уровень | Объяснение |
| Work |  | Основной каталог |
|  | Data | Содержит базу данных |
|  | Graphics | Содержит копии графических отчетов |
|  | Library | Содержит библиотеку стандартных функций |
|  | Notes | Содержит документацию |
|  | Output | Содержит копии текстовых отчетов |
|  | Scripts | Содержит main.py и дополнительные модули |

Табл. . Каталоги приложения

Листинг Скрипта

Ниже приведён список функций и docstrings каждого модуля (Табл. 4).

|  |  |
| --- | --- |
| Модуль | Функции с докстрингами |
| main.py | data\_import  Автор: Царегородцева Елизавета  Цель: импортировать данные из excel файла  Вход: глобальные переменные  Выход: измененные глобальные переменные  get\_cities  Автор: Царегородцева Елизавета  Цель: получить список городов  Вход: глобальные переменные  Выход: список  main\_window  Автор: Царегородцева Елизавета  Цель: построение главного окна  Вход: -  Выход: окно приложения |
| get\_data.py | get\_html  Автор: Царегородцева Елизавета  Цель: получить данные для парсинга сайта  Вход: ссылка на сайт  Выход: html - код  get\_names  Автор: Царегородцева Елизавета  Цель: парсинг сайта  Вход: html - код  Выход: список данных  export\_to\_excel  Автор: Царегородцева Елизавета  Цель: преобразование списка с данными в DataFrame объект и сохранение полученной базы данных в файл xlsx  Вход: список данных  Выход: файл xlsx |
| group\_diagram.py | saving  Автор: Царегородцева Елизавета  Цель: сохранение графического отчета  Вход: -  Выход: файл png  output  Автор: Царегородцева Елизавета  Цель: построение графика и вывод графика в окно приложения  Вход: индекс столбца для анализа, список данных, список, указывающий на наличие/отсутствие элемента  Выход: окно приложения с графиком |
| out\_pivot\_table.py | output  Автор: Царегородцева Елизавета  Цель: построение графика и вывод таблицы в окно приложения Вход: обьект DataFrame  Выход: окно приложения с таблицей |
| base\_statistics.py | base\_text  Автор: Царегородцева Елизавета  Цель: вывод отчета по базовой статистике для числовых переменных в окно приложения  Вход: список данных  Выход: окно приложения с таблицей  base\_num  Автор: Царегородцева Елизавета  Цель: вывод отчета по базовой статистике для числовых переменных в окно приложения  Вход: список данных  Выход: окно приложения с таблицей  saving  Автор: Царегородцева Елизавета  Цель: сохранение отчета по базовой статистике  Вход: -  Выход: файл xlsx |
| first\_tab.py | error  Автор: Мотявин Матвей  Цель: выдать сообщение об ошибке  Вход: -  Выход: окно с сообщением об ошибке  firsttab  Автор: Царегородцева Елизавета  Цель: построение и заполнение вкладки «Главное окно»  Вход: списки с данными и индексами об их наличии/отсутствии  Выход: -  sort\_column  Автор: Царегородцева Елизавета  Цель: сортировка качественных столбцов  Вход: таблица и столбец для сортировки  Выход: -  tree\_sortf  Автор: Царегородцева Елизавета  Цель: сортировка количественных столбцов  Вход: таблица и столбец для сортировки  Выход: -  inserting  Автор: Царегородцева Елизавета  Цель: заполнение таблицы данными (вкладка «Главное окно»)  Вход: -  Выход: -  creation  Автор: Царегородцева Елизавета  Цель: заполнение шапки таблицы, размещение скроллбара  Вход: -  Выход: -  table  Автор: Царегородцева Елизавета  Цель: вызов функций inserting, creation и функций сортировки  Вход: -  Выход: -  get\_city  Автор: Царегородцева Елизавета  Цель: перезаполнение таблицы в соответствие с выбранным городом  Вход: event  Выход: -  get\_analysis  Автор: Мотявин Матвей  Цель: строит окна с выбором параметров для анализа  Вход: event  Выход: вызов подходящей функции  base\_stat  Автор: Царегородцева Елизавета  Цель: проведение анализа  Вход: список данных  Выход: список с результатами анализа  sv\_table  Автор: Царегородцева Елизавета  Цель: проведение анализа  Вход: список данных  Выход: обьект DataFrame  liked  Автор: Царегородцева Елизавета  Цель: добавление данных в «Понравившееся»  Вход: event  Выход: измененные глобальные переменные  deleting  Автор: Царегородцева Елизавета  Цель: удаление элемента из таблицы  Вход: event  Выход: измененные глобальные переменные  new\_item  Автор: Царегородцева Елизавета  Цель: добавление нового элемента в тублицу  Вход: event  Выход: -  save\_changes  Автор: Царегородцева Елизавета  Цель: сохранение изменений  Вход: event  Выход: измененные глобальные переменные  changing  Автор: Царегородцева Елизавета  Цель: изменение данных в таблице  Вход: event  Выход: -  showing  Автор: Царегородцева Елизавета  Цель: открытие окна для просмотра данных  Вход: event  Выход: -  show\_all  Автор: Царегородцева Елизавета  Цель: отображение всех элементов таблицы  Вход: event  Выход: - |
| second\_tab.py | error  Автор: Мотявин Матвей  Цель: выдать сообщение об ошибке  Вход: -  Выход: окно с сообщением об ошибке  secondtab  Автор: Царегородцева Елизавета  Цель: построение и заполнение вкладки «Понравившееся»  Вход: списки с данными и индексами об их наличии/отсутствии  Выход: -  get\_analysis  Автор: Мотявин Матвей  Цель: строит окна с выбором параметров для анализа  Вход: event  Выход: вызов подходящей функции  base\_stat  Автор: Царегородцева Елизавета  Цель: проведение анализа  Вход: список данных  Выход: список с результатами анализа  sv\_table  Автор: Царегородцева Елизавета  Цель: проведение анализа  Вход: список данных  Выход: обьект DataFrame  gr\_diagram  Автор: Царегородцева Елизавета  Цель: анализ возможности построения столбчатой диаграммы  Вход: event  Выход: вызов функции или окно с ошибкой  deleting  Автор: Царегородцева Елизавета  Цель: удаление элемента из таблицы  Вход: event  Выход: измененные глобальные переменные  sort\_column  Автор: Царегородцева Елизавета  Цель: сортировка качественных столбцов  Вход: таблица и столбец для сортировки  Выход: -  tree\_sortf  Автор: Царегородцева Елизавета  Цель: сортировка количественных столбцов  Вход: таблица и столбец для сортировки  Выход: -  inserting  Автор: Царегородцева Елизавета  Цель: заполнение таблицы данными (вкладка «Главное окно»)  Вход: -  Выход: -  creation  Автор: Царегородцева Елизавета  Цель: заполнение шапки таблицы, размещение скроллбара  Вход: -  Выход: -  saving  Автор: Царегородцева Елизавета  Цель: сохранение новой таблицы  Вход: event  Выход: excel файл  showing  Автор: Царегородцева Елизавета  Цель: открытие окна для просмотра данных  Вход: event  Выход: -  histogram  Автор: Мотявин Матвей  Цель: построение гистограммы  Вход: названия столбцов  Выход: -  save\_plot  Автор: Мотявин Матвей  Цель: сохранение графического отчета  Вход: название фигуры  Выход: файл png  boxgraph  Автор: Мотявин Матвей  Цель: построение диаграммы Бокса-Вискера  Вход: названия столбцов  Выход: -  check  Автор: Мотявин Матвей  Цель: проверить все ли данные в выбранном стоблце идентичны  Вход: список  Выход: список  scatter  Автор: Мотявин Матвей  Цель: построение диаграммы рассеивания  Вход: названия столбцов  Выход: - |

Табл. . Список функция и docstrings

**main.py**

"""Cтроит главное окно"""

from tkinter import Tk, ttk

import numpy as np

import pandas as pd

import get\_data

import first\_tab

import second\_tab

def data\_import(landmarks, contacts, informations):

"""Автор: Царегородцева Елизавета

Цель: импортировать данные из excel файла

Вход: глобальные переменные

Выход: измененные глобальные переменные"""

data\_open = pd.read\_excel('../Data/data.xlsx',

sheet\_name='Основная таблица')

landmarks = data\_open.to\_numpy()[:][:].tolist()

for landmark in landmarks:

del landmark[0]

data\_open = pd.read\_excel('../Data/data.xlsx', sheet\_name='Контакты')

contacts = data\_open.to\_numpy()[:][:].tolist()

for contact in contacts:

del contact[0]

data\_open = pd.read\_excel('../Data/data.xlsx',

sheet\_name='Дополнительная информация')

informations = data\_open.to\_numpy()[:][:].tolist()

for information in informations:

del information[0]

return landmarks, contacts, informations

def get\_cities(landmarks):

"""Автор: Царегородцева Елизавета

Цель: получить список городов

Вход: глобальные переменные

Выход: список"""

cities = (np.array(landmarks))[:, 3]

cities\_m = []

for i in range(cities.size):

if cities[i] not in cities\_m:

cities\_m.append(cities[i])

cities = np.array(cities\_m)

cities.sort()

cities\_m.clear()

return cities

def main\_window():

"""Автор: Царегородцева Елизавета

Цель: построение главного окна

Вход: -

Выход: окно приложения"""

root = Tk()

root.title('Достопримечательности России')

root.resizable(False, False)

root.width = root.winfo\_screenwidth() // 2

root.height = root.winfo\_screenheight() // 2

root.geometry('+{}+{}'.format(root.width-400, root.height-400))

notebook = ttk.Notebook(root, width=1000, height=700)

tab\_1 = ttk.Frame(root)

tab\_2 = ttk.Frame(root)

notebook.add(tab\_1, text="Главное окно")

notebook.add(tab\_2, text="Понравившееся")

liked\_list = []

first\_tab.firsttab(tab\_1, tab\_2, landmarks, cities, liked\_list, table\_ind,

new\_ind, count\_1, count\_2)

second\_tab.secondtab(tab\_2, liked\_list, table\_ind, new\_ind, count\_2)

notebook.pack(fill='both', expand=True)

root.mainloop()

get\_data.get\_html("https://autotravel.ru/top100.php")

landmarks, contacts, informations = [], [], []

landmarks, contacts, informations = data\_import(landmarks, contacts,

informations)

for i in range(len(landmarks)):

landmarks[i][3] = contacts[i]

landmarks[i][4] = informations[i]

df = pd.DataFrame.from\_dict(landmarks)

for i in range(len(landmarks)):

landmarks[i][3] = landmarks[i][2]

landmarks[i][2] = landmarks[i][1]

landmarks[i][1] = landmarks[i][0]

landmarks[i][0] = contacts[i][0]

landmarks[i][4] = contacts[i][1]

landmarks[i].append(contacts[i][2])

landmarks[i].append(informations[i][0])

landmarks[i].append(informations[i][1])

landmarks[i].append(informations[i][2])

landmarks[i].append(informations[i][3])

cities = get\_cities(landmarks)

contacts.clear()

informations.clear()

new\_ind = []

count\_1, count\_2 = [], []

for landmark in landmarks:

count\_1.append(0)

count\_2.append(0)

table\_ind = np.array(['I001', 'I002', 'I003', 'I004', 'I005', 'I006', 'I007',

'I008', 'I009', 'I00A', 'I00B', 'I00C', 'I00D', 'I00E',

'I00F', 'I010', 'I011', 'I012', 'I013', 'I014', 'I015',

'I016', 'I017', 'I018', 'I019', 'I01A', 'I01B', 'I01C',

'I01D', 'I01E', 'I01F', 'I020', 'I021', 'I022', 'I023',

'I024', 'I025', 'I026', 'I027', 'I028', 'I029', 'I02A',

'I02B', 'I02C', 'I02D', 'I02E', 'I02F', 'I030', 'I031',

'I032', 'I033', 'I034', 'I035', 'I036', 'I037', 'I038',

'I039', 'I03A', 'I03B', 'I03C', 'I03D', 'I03E', 'I03F',

'I040', 'I041', 'I042', 'I043', 'I044', 'I045', 'I046',

'I047', 'I048', 'I049', 'I04A', 'I04B', 'I04C', 'I04D',

'I04E', 'I04F', 'I050', 'I051', 'I052', 'I053', 'I054',

'I055', 'I056', 'I057', 'I058', 'I059', 'I05A', 'I05B',

'I05C', 'I05D', 'I05E', 'I05F', 'I060', 'I061', 'I062',

'I063', 'I064', 'I065', 'I066', 'I067', 'I068', 'I069'])

main\_window()

**get\_data.py**

"""Парсинг сайта."""

import re

import random

import requests

import pandas as pd

from pandas import ExcelWriter as ew

from bs4 import BeautifulSoup

def get\_html(url):

"""Автор: Царегородцева Елизавета

Цель: получить данные для парсинга сайта

Вход: ссылка на сайт

Выход: html - код"""

response = requests.get(url)

response = response.text

landmarks = []

contacts = []

information = []

get\_names(response, landmarks, contacts, information)

export\_to\_excel(landmarks, contacts, information)

def get\_names(data, landmarks, contacts, information):

"""Автор: Царегородцева Елизавета

Цель: парсинг сайта

Вход: html - код

Выход: список данных"""

soup = BeautifulSoup(data, 'html.parser')

items = soup.find('div', {'class': 'autocard-masonry'}

).find\_all('div', {'class': 'autocard'})

for item in items:

lm\_price = random.randint(200, 400)

lm\_link = 'https://autotravel.ru' + item.find('a').get('href')

lm\_title = item.find('a').text

places = item.find('p', {'class': 'mb-2 text-muted t0'}).find\_all('a')

i = 0

for place in places:

if i == 0:

lm\_region = place.text

i += 1

else:

lm\_city = place.text

lm\_info = item.find('div', {'class': 'tj t0'}).text

if lm\_info.partition('Режим работы:'):

lm\_info = lm\_info.partition('Режим работы:')

lm\_schedule = lm\_info[2]

lm\_info = list(lm\_info)

del lm\_info[1]

del lm\_info[1]

lm\_info = ''.join(lm\_info)

if lm\_info.partition('Адрес:'):

lm\_info = lm\_info.partition('Адрес:')

lm\_ad = lm\_info[2]

lm\_info = list(lm\_info)

del lm\_info[1]

del lm\_info[1]

lm\_info = ''.join(lm\_info)

lm\_rating = float(item.find('strong').text)

if lm\_ad.partition('тел.'):

lm\_ad = lm\_ad.partition('тел.')

lm\_tel = lm\_ad[2]

lm\_ad = list(lm\_ad)

del lm\_ad[1]

del lm\_ad[1]

lm\_ad = ''.join(lm\_ad)

lm\_ad = re.sub(r"[^\w]\*$", '', lm\_ad)

else:

if lm\_ad.partition('Тел.'):

lm\_ad = lm\_ad.partition('Тел.')

lm\_tel = lm\_ad[2]

lm\_ad = list(lm\_ad)

del lm\_ad[1]

del lm\_ad[1]

lm\_ad = ''.join(lm\_ad)

lm\_ad = re.sub(r"[^\w]\*$", '', lm\_ad)

landmarks.append({

'lm\_title': lm\_title,

'lm\_region': lm\_region,

'lm\_city': lm\_city,

'contacts': 'Контакты',

'information': 'Дополнительная информация'

})

contacts.append({

'lm\_link': lm\_link,

'lm\_contacts\_ad': lm\_ad,

'lm\_contacts\_tel': lm\_tel

})

information.append({

'lm\_schedule': lm\_schedule,

'lm\_rating': lm\_rating,

'lm\_price': lm\_price,

'lm\_info': lm\_info

})

return landmarks, contacts, information

def export\_to\_excel(landmarks, contacts, information):

"""Автор: Царегородцева Елизавета

Цель: преобразование списка с данными в DataFrame объект и сохранение полученной базы данных в файл xlsx

Вход: список данных

Выход: файл xlsx"""

df1 = pd.DataFrame.from\_dict(landmarks)

df2 = pd.DataFrame.from\_dict(contacts)

df3 = pd.DataFrame.from\_dict(information)

landmarks.clear()

contacts.clear()

information.clear()

wrt = ew('../Data/data.xlsx', engine='xlsxwriter')

df1.to\_excel(wrt, 'Основная таблица')

df2.to\_excel(wrt, 'Контакты')

df3.to\_excel(wrt, 'Дополнительная информация')

wrt.save()

**out\_pivot\_table.py**

"""Вывод сводной таблицы в окно"""

import tkinter as tk

def output(data):

"""Автор: Царегородцева Елизавета

Цель: построение графика и вывод таблицы в окно приложения

Вход: обьект DataFrame

Выход: окно приложения с таблицей"""

table = tk.Toplevel()

table.title('Сводная таблица')

table.geometry('1000x600+400+300')

table.resizable(False, False)

text = tk.Text(table, state='normal', width=120,

height=50, wrap='word')

text.grid(row=1, column=0, padx=5, pady=5)

text.insert('end', data)

table.mainloop()

**group\_diagram.py**

"""Построение и сохранение столбчатой диаграммы"""

import tkinter as tk

from tkinter import filedialog as fd

import matplotlib.pyplot as plt

import matplotlib as mpl

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

def output(index, liked\_list, count\_2):

"""Автор: Царегородцева Елизавета

Цель: построение графика и вывод графика в окно приложения

Вход: индекс столбца для анализа, список данных, список, указывающий на наличие/отсутствие элемента

Выход: окно приложения с графиком"""

def saving():

"""Автор: Царегородцева Елизавета

Цель: сохранение графического отчета

Вход: -

Выход: файл png"""

savefile = fd.asksaveasfilename(filetypes=(("PNG Files", "\*.png"),

("All files", "\*.\*")))

if savefile:

fig.savefig(savefile + ".png")

data\_names = []

data\_values1 = []

data\_values2 = []

for i in range(len(liked\_list)):

count = 0

if count\_2[i] == 1:

if len(data\_names) > 0:

if liked\_list[i][index] in data\_names:

count = 1

else:

data\_names.append(liked\_list[i][index])

count = 1

if count == 0:

data\_names.append(liked\_list[i][index])

for i in range(len(data\_names)):

index1 = 0

index2 = 0

for j in range(len(liked\_list)):

if (liked\_list[j][index] == data\_names[i]) and (count\_2[j] == 1):

if 'кремль' in liked\_list[j][1]:

index1 += 1

elif 'Кремль' in liked\_list[j][1]:

index1 += 1

else:

index2 += 1

data\_values1.append(index1)

data\_values2.append(index2)

dpi = 90

fig = plt.figure(dpi=dpi, figsize=(512/dpi, 500/dpi))

mpl.rcParams.update({'font.size': 10})

ax = plt.axes()

ax.yaxis.grid(True, zorder=1)

xs = range(len(data\_names))

plt.bar([x + 0.05 for x in xs], data\_values1,

width=0.2, color='red', alpha=0.7, label='Кремль',

zorder=2)

plt.bar([x + 0.3 for x in xs], data\_values2,

width=0.2, color='blue', alpha=0.7, label='Остальное',

zorder=2)

plt.xticks(xs, data\_names)

data\_names = ax.set\_xticklabels(data\_names,

fontsize=10,

rotation=25,

verticalalignment='center')

plt.legend(loc='upper right')

diag = tk.Toplevel()

diag.title('Столбчатая диаграмма')

diag.geometry('500x800')

diag.resizable(False, False)

FigureCanvasTkAgg(fig, diag).get\_tk\_widget().pack(side=tk.LEFT,

fill=tk.BOTH)

button\_save = tk.Button(diag, text="Сохранить", width=20,

command=saving)

button\_save.place(y=20)

diag.mainloop()

**base\_statistics.py**

"""Вывод базовой статистики в окно и сохранение"""

from tkinter import ttk

import tkinter as tk

from tkinter import filedialog as fd

import pandas as pd

def base\_text(output):

"""Автор: Царегородцева Елизавета

Цель: вывод отчета по базовой статистике для текстовых переменных в окно приложения

Вход: список данных

Выход: - """

def saving():

"""Автор: Царегородцева Елизавета

Цель: сохранение отчета по базовой статистике

Вход: -

Выход: файл xlsx"""

savefile = fd.asksaveasfilename(filetypes=(("Excel files", "\*.xlsx"),

("All files", "\*.\*")))

if savefile:

df = pd.DataFrame.from\_dict(output)

df.to\_excel(savefile + ".xlsx", index=False,

sheet\_name="Базовая статистика")

output.clear()

base = tk.Toplevel()

base.title('Базовая статистика')

base.geometry('600x400+400+300')

base.resizable(False, False)

columns = ("Значение", "Частота", "Процент от общего числа")

tree\_base = ttk.Treeview(base, columns=columns, height=20)

tree\_base.heading("0", text="Значение")

tree\_base.heading("1", text="Частота")

tree\_base.heading("2", text="Процент от общего числа")

tree\_base['show'] = 'headings'

tree\_base.column("0", width=150, stretch=False, anchor='c')

tree\_base.column("1", width=150, stretch=False, anchor='c')

tree\_base.column("2", width=200, stretch=False, anchor='c')

tree\_base.place(x=0, y=40)

scrll = tk.Scrollbar(base, orient="vertical",

command=tree\_base.yview)

scrll.place(x=502, y=40, height=360)

tree\_base.configure(yscrollcommand=scrll.set)

i = int(0)

for i in range(len(output)):

tree\_base.insert('', 'end', values=(output[i]["Значение"],

output[i]["Частота"],

output[i]["Процент от общего числа"]))

button\_save = tk.Button(base, text="Сохранить",

command=saving)

button\_save.grid(column=0, row=0, pady=10, padx=10)

base.mainloop()

def base\_num(output):

"""Автор: Царегородцева Елизавета

Цель: вывод отчета по базовой статистике для числовых переменных в окно приложения

Вход: список данных

Выход: - """

def saving():

"""Автор: Царегородцева Елизавета

Цель: сохранение отчета по базовой статистике

Вход: -

Выход: файл xlsx"""

savefile = fd.asksaveasfilename(filetypes=(("Excel files", "\*.xlsx"),

("All files", "\*.\*")))

if savefile:

df = pd.DataFrame.from\_dict(output)

df.to\_excel(savefile + ".xlsx", index=False,

sheet\_name="Базовая статистика")

output.clear()

base = tk.Toplevel()

base.title('Базовая статистика')

base.geometry('900x400+400+300')

base.resizable(False, False)

columns = ("Переменная", "Максимум", "Минимум", "Среднее арифметическое",

"Выборочная дисперсия",

"Стандартное отклонение")

tree\_base = ttk.Treeview(base, columns=columns, height=20)

tree\_base.heading("0", text="Переменная")

tree\_base.heading("1", text="Максимум")

tree\_base.heading("2", text="Минимум")

tree\_base.heading("3", text="Среднее арифметическое")

tree\_base.heading("4", text="Выборочная дисперсия")

tree\_base.heading("5", text="Стандартное отклонение")

tree\_base['show'] = 'headings'

tree\_base.column("0", width=100, stretch=False, anchor='c')

tree\_base.column("1", width=80, stretch=False, anchor='c')

tree\_base.column("2", width=80, stretch=False, anchor='c')

tree\_base.column("3", width=150, stretch=False, anchor='c')

tree\_base.column("4", width=150, stretch=False, anchor='c')

tree\_base.column("5", width=150, stretch=False, anchor='c')

tree\_base.place(x=0, y=40)

i = int(0)

for i in range(len(output)):

tree\_base.insert('', 'end', values=(output[i]["Переменная"],

output[i]["Максимум"],

output[i]["Минимум"],

output[i]["Среднее арифметическое"],

output[i]["Выборочная дисперсия"],

output[i]["Стандартное отклонение"]))

button\_save = tk.Button(base, text="Сохранить",

command=saving)

button\_save.grid(column=0, row=0, pady=10, padx=10)

base.mainloop()

**first\_tab.py**

"""Строит вкладку 'Главное окно'."""

from tkinter import ttk

import tkinter as tk

from tkinter import messagebox as mb

import pandas as pd

import numpy as np

import base\_statistics

import out\_pivot\_table

import second\_tab

def error():

"""Автор: Мотявин Матвей

Цель: выдать сообщение об ошибке

Вход: -

Выход: окно с сообщением об ошибке """

mb.showinfo('Ошибка',

'Невозможно произвести анализ с данными параметрами.')

def firsttab(tab\_1, tab\_2, landmarks, cities, liked\_list, table\_ind, new\_ind, count\_1, count\_2):

"""Автор: Царегородцева Елизавета

Цель: построение и заполнение вкладки «Главное окно»

Вход: списки с данными и индексами об их наличии/отсутствии

Выход: -"""

def sort\_column(tv, col, reverse):

"""Автор: Царегородцева Елизавета

Цель: сортировка качественных столбцов

Вход: таблица и столбец для сортировки

Выход: -"""

l = [(tv.set(k, col), k) for k in tv.get\_children('')]

l.sort(reverse=reverse)

for index, (val, k) in enumerate(l):

tv.move(k, '', index)

tv.heading(col, text=col, command=lambda \_col=col:

sort\_column(tv, \_col, not reverse))

def tree\_sortf(tv, col, reverse):

"""Автор: Царегородцева Елизавета

Цель: сортировка количественных столбцов

Вход: таблица и столбец для сортировки

Выход: -"""

l = [(tv.set(k, col), k) for k in tv.get\_children('')]

l.sort(key=lambda rate: float(rate[0]), reverse=reverse)

for index, (val, k) in enumerate(l):

tv.move(k, '', index)

tv.heading(col, command=lambda: tree\_sortf(tv, col, not reverse))

def inserting():

"""Автор: Царегородцева Елизавета

Цель: заполнение таблицы данными (вкладка «Главное окно»)

Вход: -

Выход: - """

for i in tree.get\_children():

tree.delete(i)

for i in range(len(landmarks)):

if count\_1[i] == 0:

tree.insert('', 'end', values=(landmarks[i][1],

landmarks[i][3],

landmarks[i][2],

landmarks[i][7],

landmarks[i][0]))

def creation():

"""Автор: Царегородцева Елизавета

Цель: заполнение шапки таблицы, размещение скроллбара

Вход: -

Выход: -"""

tree['show'] = 'headings'

tree.column("0", width=150, stretch=False, anchor='c')

tree.column("1", width=150, stretch=False, anchor='c')

tree.column("2", width=150, stretch=False, anchor='c')

tree.column("3", width=140, stretch=False, anchor='c')

tree.column("4", width=300, stretch=False, anchor='c')

tree.heading("4", text="Ссылка")

tree.place(x=30, y=220)

scrll = ttk.Scrollbar(tab\_1, orient="vertical",

command=tree.yview)

scrll.place(x=922, y=220, height=430)

tree.configure(yscrollcommand=scrll.set)

def table():

"""Автор: Царегородцева Елизавета

Цель: вызов функций inserting, creation и функций сортировки

Вход: -

Выход: -"""

creation()

inserting()

columns\_sort = ("Название", "Город", "Регион")

column = ("Рейтинг",)

for col in columns\_sort:

tree.heading(col, text=col, command=lambda \_col=col:

sort\_column(tree, \_col, False))

for col in column:

tree.heading(col, text=col, command=lambda:

tree\_sortf(tree, col, False))

def get\_city():

"""Автор: Царегородцева Елизавета

Цель: перезаполнение таблицы в соответствие с выбранным городом

Вход: event

Выход: -"""

choosen\_city = combobox\_1.get()

for i in tree.get\_children():

tree.delete(i)

for i in range(len(landmarks)):

if landmarks[i][3] == choosen\_city:

tree.insert('', 'end', values=(landmarks[i][1],

landmarks[i][3],

landmarks[i][2],

landmarks[i][7],

landmarks[i][0]))

def get\_analysis():

"""Автор: Мотявин Матвей

Цель: строит окна с выбором параметров для анализа

Вход: event

Выход: вызов подходящей функции"""

def base\_stat():

"""Автор: Царегородцева Елизавета

Цель: проведение анализа

Вход: список данных

Выход: список с результатами анализа"""

amount = 0

for i in range(len(count\_1)):

if count\_1[i] == 0:

amount += 1

if amount == 1:

mb.showinfo('Ошибка', 'Таблица пуста.')

else:

if var.get() == 0:

error()

elif var.get() == 3:

error()

elif var.get() == 1:

output = []

count = 1

index = 0

for i in range(len(count\_1)-1, -1, -1):

if count\_1[i] == 0:

index = i

for i in range(index+1, len(landmarks)):

if (landmarks[index][3] == landmarks[i][3] and count\_1[i] == 0):

count += 1

output.append({

'Значение': landmarks[index][3],

'Частота': count,

'Процент от общего числа':

str(count/amount\*100) + "%",

})

for i in range(index+1, len(landmarks)):

repeat = 0

for rep in range(len(output)):

if (landmarks[i][3] == output[rep]['Значение'] and count\_1[i] == 0):

repeat += 1

if repeat == 0:

count = 0

for j in range(index+1, len(landmarks)):

if (landmarks[i][3] == landmarks[j][3] and count\_1[i] == 0):

count += 1

if count > 0:

output.append({

'Значение': landmarks[i][3],

'Частота': count,

'Процент от общего числа':

str(count/amount\*100) + "%"})

elif var.get() == 2:

output = []

count = 1

index = 0

for i in range(len(count\_1)-1, -1, -1):

if count\_1[i] == 0:

index = i

for i in range(index+1, len(landmarks)):

if (landmarks[index][2] == landmarks[i][2] and count\_1[i] == 0):

count += 1

output.append({

'Значение': landmarks[index][2],

'Частота': count,

'Процент от общего числа': str(count) + "%",

})

for i in range(index+1, len(landmarks)):

repeat = 0

for rep in range(len(output)):

if (landmarks[i][2] == output[rep]['Значение'] and count\_1[i] == 0):

repeat += 1

if repeat == 0:

count = 0

for j in range(index+1, len(landmarks)):

if (landmarks[i][2] == landmarks[j][2] and count\_1[i] == 0):

count += 1

if count > 0:

output.append({

'Значение': landmarks[i][2],

'Частота': count,

'Процент от общего числа': str(count/amount\*100) + "%"

})

elif var.get() == 4:

dfcolumns = ['lm\_link', 'lm\_title', 'lm\_region',

'lm\_city', 'lm\_contacts\_ad',

'lm\_contacts\_tel', 'lm\_schedule',

'lm\_rating', 'lm\_price', 'lm\_info']

land = []

output = []

j = int(0)

for i in range(len(landmarks)):

land.append(0)

for i in range(len(landmarks)):

if count\_1[i] == 0:

land[j] = dict(zip(dfcolumns, landmarks[i]))

j += 1

df = pd.DataFrame.from\_dict(land)

output.append({

'Переменная': 'Рейтинг',

'Максимум': df['lm\_rating'].max(),

'Минимум': df['lm\_rating'].min(),

'Среднее арифметическое': df['lm\_rating'].mean(),

'Выборочная дисперсия': df['lm\_rating'].std(0),

'Стандартное отклонение': df['lm\_rating'].var(ddof=1),

})

elif var.get() == 5:

dfcolumns = ['lm\_link', 'lm\_title', 'lm\_region',

'lm\_city', 'lm\_contacts\_ad',

'lm\_contacts\_tel', 'lm\_schedule',

'lm\_rating', 'lm\_price', 'lm\_info']

land = []

output = []

j = int(0)

for i in range(len(landmarks)):

land.append(0)

for i in range(len(landmarks)):

if count\_1[i] == 0:

land[j] = dict(zip(dfcolumns, landmarks[i]))

j += 1

df = pd.DataFrame.from\_dict(land)

output.append({

'Переменная': 'Цена',

'Максимум': df['lm\_price'].max(),

'Минимум': df['lm\_price'].min(),

'Среднее арифметическое': df['lm\_price'].mean(),

'Выборочная дисперсия': df['lm\_price'].std(0),

'Стандартное отклонение': df['lm\_price'].var(ddof=1),

})

dialog1.destroy()

if (var.get() == 1 or var.get() == 2):

base\_statistics.base\_text(output)

if (var.get() == 4 or var.get() == 5):

base\_statistics.base\_num(output)

def sv\_table():

"""Автор: Царегородцева Елизавета

Цель: проведение анализа

Вход: список данных

Выход: обьект DataFrame"""

amount = 0

for i in range(len(count\_1)):

if count\_1[i] == 0:

amount += 1

if amount == 1:

mb.showinfo('Ошибка', 'Таблица пуста.')

else:

if entry\_quality1.get() == 'Регион':

if entry\_quality2.get() == 'Город':

if agreg.get() == 'Цена':

new\_list = []

for i in range(len(count\_1)):

if count\_1[i] == 0:

new\_list.append({'Регион': landmarks[i][2],

'Город': landmarks[i][3],

'Цена': landmarks[i][8]})

df = pd.DataFrame.from\_dict(new\_list)

data = pd.pivot\_table(df, index=['Регион', 'Город'],

values=['Цена'],

aggfunc=np.sum)

else:

new\_list = []

for i in range(len(count\_1)):

if count\_1[i] == 0:

new\_list.append({'Регион': landmarks[i][2],

'Город': landmarks[i][3],

'Количество достопримечательностей':

int(1)})

df = pd.DataFrame.from\_dict(new\_list)

data = pd.pivot\_table(df, index=['Регион', 'Город'],

values=['Количество достопримечательностей'],

aggfunc=np.sum)

dialog2.destroy()

out\_pivot\_table.output(data)

else:

error()

choosen\_analysis = combobox\_2.get()

if choosen\_analysis == "Базовая статистика":

dialog1 = tk.Toplevel()

dialog1.title('Параметры базовой статистики')

dialog1.geometry('300x300+400+300')

dialog1.resizable(False, False)

var = tk.IntVar()

var.set(0)

title = ttk.Radiobutton(dialog1, text='Название', variable=var, value=0)

title.place(x=50, y=30)

city = ttk.Radiobutton(dialog1, text='Город', variable=var, value=1)

city.place(x=50, y=60)

region = ttk.Radiobutton(dialog1, text='Регион', variable=var, value=2)

region.place(x=50, y=90)

link = ttk.Radiobutton(dialog1, text='Ссылка', variable=var, value=3)

link.place(x=50, y=120)

rating = ttk.Radiobutton(dialog1, text='Рейтинг', variable=var, value=4)

rating.place(x=50, y=150)

price = ttk.Radiobutton(dialog1, text='Цена', variable=var, value=5)

price.place(x=50, y=180)

btn\_ok = tk.Button(dialog1, text='Начать анализ', command=base\_stat)

btn\_ok.place(x=50, y=250)

dialog1.mainloop()

elif choosen\_analysis == 'Сводная таблица':

dialog2 = tk.Toplevel()

dialog2.title('Параметры сводной таблицы')

dialog2.geometry('400x220+400+300')

dialog2.resizable(False, False)

label\_quality1 = tk.Label(dialog2, text='Качественный параметр 1')

label\_quality1.place(x=50, y=50)

label\_quality2 = tk.Label(dialog2, text='Качественный параметр 2')

label\_quality2.place(x=50, y=80)

label\_numerical = tk.Label(dialog2, text='Метод агрегации')

label\_numerical.place(x=50, y=110)

entry\_quality1 = ttk.Combobox(dialog2, values=['Название', 'Город',

'Регион', 'Телефон',

'Ссылка'])

entry\_quality1.current(0)

entry\_quality1.place(x=200, y=50)

entry\_quality2 = ttk.Combobox(dialog2, values=['Название', 'Город',

'Регион', 'Телефон',

'Ссылка'])

entry\_quality2.current(0)

entry\_quality2.place(x=200, y=80)

agreg = ttk.Combobox(dialog2, values=['Цена',

'Количество достопримечательностей'])

agreg.current(0)

agreg.place(x=200, y=110)

dialog2.btn\_ok = tk.Button(dialog2, text='Начать анализ',

command=sv\_table)

dialog2.btn\_ok.place(x=200, y=170)

dialog2.mainloop()

else:

mb.showinfo('Ошибка',

'Эта функция возможна только в разделе "Понравившееся".')

def liked():

"""Автор: Царегородцева Елизавета

Цель: добавление данных в «Понравившееся»

Вход: event

Выход: измененные глобальные переменные"""

try:

item = tree.selection()[0]

k = int(0)

for i in range(table\_ind.size):

if item == table\_ind[i]:

index = i

if (len(liked\_list) == 0 and 1 not in count\_2):

liked\_list.append(landmarks[index])

new\_ind.append(str(table\_ind[0]))

k = 1

count\_2[0] = 1

else:

rep = -1

for j in range(len(liked\_list)):

if landmarks[index][1] == liked\_list[j][1]:

rep = j

if rep > -1:

if count\_2[rep] == 1:

mb.showerror("Ошибка",

"Этот элемент уже добавлен в список.")

k = 1

else:

count\_2[rep] = 1

k = 1

if k == 0:

liked\_list.append(landmarks[index])

new\_ind.append(str(table\_ind[len(liked\_list)-1]))

count\_2[len(liked\_list)-1] = 1

second\_tab.secondtab(tab\_2, liked\_list, table\_ind, new\_ind,

count\_2)

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

def deleting():

"""Автор: Царегородцева Елизавета

Цель: удаление элемента из таблицы

Вход: event

Выход: измененные глобальные переменные"""

try:

item = tree.selection()[0]

tree.delete(item)

for i in range(len(table\_ind)-1):

if item == table\_ind[i]:

count\_1[i] -= 1

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

def new\_item():

"""Автор: Царегородцева Елизавета

Цель: добавление нового элемента в тублицу

Вход: event

Выход: -"""

def save\_changes():

"""Автор: Царегородцева Елизавета

Цель: сохранение изменений

Вход: event

Выход: измененные глобальные переменные"""

index = len(landmarks)

count\_1.append(0)

count\_2.append(0)

new = []

new.append(link\_e.get())

new.append(title\_e.get())

new.append(region\_e.get())

new.append(city\_e.get())

new.append(ad\_e.get())

new.append(tel\_e.get())

new.append(schedule\_e.get())

if rating\_e.get() == '':

mb.showerror('Ошибка',

'Обязательно заполните поле "Рейтинг"!')

else:

if rating\_e.get().isdigit():

new.append(rating\_e.get())

new.append(price\_e.get())

new.append(info\_e.get(1.0, 'end'))

new[9] = new[9].replace("\n", "")

landmarks.append(new)

showinfo.destroy()

tree.insert('', 'end', values=(landmarks[index][1],

landmarks[index][3],

landmarks[index][2],

landmarks[index][7],

landmarks[index][0]))

else:

mb.showerror('Ошибка',

'В поле "Рейтинг" должно быть числовое значение!')

try:

showinfo = tk.Toplevel()

showinfo.title("Добавление новой строки")

showinfo.geometry('500x610+450+140')

showinfo.resizable(False, False)

title = tk.Label(showinfo, text="Название:")

city = tk.Label(showinfo, text="Город:")

region = tk.Label(showinfo, text="Регион:")

link = tk.Label(showinfo, text="Ссылка:")

tel = tk.Label(showinfo, text="Телефон:")

ad = tk.Label(showinfo, text="Адрес:")

rating = tk.Label(showinfo, text="Рейтинг:")

schedule = tk.Label(showinfo, text="Время работы:")

price = tk.Label(showinfo, text="Цена:")

info = tk.Label(showinfo, text="Описание:")

title.grid(row=0, column=0, sticky="w")

city.grid(row=1, column=0, sticky="w")

region.grid(row=2, column=0, sticky="w")

link.grid(row=3, column=0, sticky="w")

tel.grid(row=4, column=0, sticky="w")

ad.grid(row=5, column=0, sticky="w")

rating.grid(row=6, column=0, sticky="w")

schedule.grid(row=7, column=0, sticky="w")

price.grid(row=8, column=0, sticky="w")

info.grid(row=9, column=0, sticky="w")

title\_e = tk.Entry(showinfo, width=40)

city\_e = tk.Entry(showinfo, width=40)

region\_e = tk.Entry(showinfo, width=40)

link\_e = tk.Entry(showinfo, width=40)

tel\_e = tk.Entry(showinfo, width=40)

ad\_e = tk.Entry(showinfo, width=40)

rating\_e = tk.Entry(showinfo, width=40)

schedule\_e = tk.Entry(showinfo, width=40)

price\_e = tk.Entry(showinfo, width=40)

info\_e = tk.Text(showinfo, state='normal', width=30,

height=15, wrap='word')

title\_e.grid(row=0, column=1, padx=5, pady=5)

city\_e.grid(row=1, column=1, padx=5, pady=5)

region\_e.grid(row=2, column=1, padx=5, pady=5)

link\_e.grid(row=3, column=1, padx=5, pady=5)

tel\_e.grid(row=4, column=1, padx=5, pady=5)

ad\_e.grid(row=5, column=1, padx=5, pady=5)

rating\_e.grid(row=6, column=1, padx=5, pady=5)

schedule\_e.grid(row=7, column=1, padx=5, pady=5)

price\_e.grid(row=8, column=1, padx=5, pady=5)

info\_e.grid(row=9, column=1, padx=5, pady=5)

button\_ok = ttk.Button(showinfo, text="Добавить",

command=save\_changes)

button\_ok.grid(row=10, column=1, padx=5, pady=5)

showinfo.mainloop()

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

def changing():

"""Автор: Царегородцева Елизавета

Цель: изменение данных в таблице

Вход: event

Выход: -"""

def save\_changes():

"""Автор: Царегородцева Елизавета

Цель: сохранение изменений

Вход: event

Выход: измененные глобальные переменные"""

landmarks[index][5] = tel\_e.get()

landmarks[index][6] = ad\_e.get()

landmarks[index][8] = price\_e.get()

showinfo.destroy()

try:

item = tree.selection()[0]

for i in range(len(table\_ind)):

if item == table\_ind[i]:

index = i

showinfo = tk.Toplevel()

showinfo.title("Изменение данных")

showinfo.geometry('500x610+450+140')

showinfo.resizable(False, False)

title = tk.Label(showinfo, text="Название:")

city = tk.Label(showinfo, text="Город:")

region = tk.Label(showinfo, text="Регион:")

link = tk.Label(showinfo, text="Ссылка:")

tel = tk.Label(showinfo, text="Телефон:")

ad = tk.Label(showinfo, text="Адрес:")

rating = tk.Label(showinfo, text="Рейтинг:")

schedule = tk.Label(showinfo, text="Время работы:")

price = tk.Label(showinfo, text="Цена:")

info = tk.Label(showinfo, text="Описание:")

title.grid(row=0, column=0, sticky="w")

city.grid(row=1, column=0, sticky="w")

region.grid(row=2, column=0, sticky="w")

link.grid(row=3, column=0, sticky="w")

tel.grid(row=4, column=0, sticky="w")

ad.grid(row=5, column=0, sticky="w")

rating.grid(row=6, column=0, sticky="w")

schedule.grid(row=7, column=0, sticky="w")

price.grid(row=8, column=0, sticky="w")

info.grid(row=9, column=0, sticky="w")

title\_e = tk.Entry(showinfo, width=40)

city\_e = tk.Entry(showinfo, width=40)

region\_e = tk.Entry(showinfo, width=40)

link\_e = tk.Entry(showinfo, width=40)

tel\_e = tk.Entry(showinfo, width=40)

ad\_e = tk.Entry(showinfo, width=40)

rating\_e = tk.Entry(showinfo, width=40)

schedule\_e = tk.Entry(showinfo, width=40)

price\_e = tk.Entry(showinfo, width=40)

info\_e = tk.Text(showinfo, state='disabled', width=30,

height=15, wrap='word')

title\_e.grid(row=0, column=1, padx=5, pady=5)

city\_e.grid(row=1, column=1, padx=5, pady=5)

region\_e.grid(row=2, column=1, padx=5, pady=5)

link\_e.grid(row=3, column=1, padx=5, pady=5)

tel\_e.grid(row=4, column=1, padx=5, pady=5)

ad\_e.grid(row=5, column=1, padx=5, pady=5)

rating\_e.grid(row=6, column=1, padx=5, pady=5)

schedule\_e.grid(row=7, column=1, padx=5, pady=5)

price\_e.grid(row=8, column=1, padx=5, pady=5)

info\_e.grid(row=9, column=1, padx=5, pady=5)

title\_e.insert(0, landmarks[index][1])

city\_e.insert(0, landmarks[index][3])

region\_e.insert(0, landmarks[index][2])

link\_e.insert(0, landmarks[index][0])

tel\_e.insert(0, landmarks[index][5])

ad\_e.insert(0, landmarks[index][4])

rating\_e.insert(0, landmarks[index][7])

schedule\_e.insert(0, landmarks[index][6])

price\_e.insert(0, landmarks[index][8])

info\_e.configure(state='normal')

info\_e.insert('end', landmarks[index][9])

info\_e.configure(state='disabled')

title\_e['state'] = 'disabled'

city\_e['state'] = 'disabled'

region\_e['state'] = 'disabled'

rating\_e['state'] = 'disabled'

link\_e['state'] = 'disabled'

button\_ok = ttk.Button(showinfo, text="Сохранить изменения",

command=save\_changes)

button\_ok.grid(row=10, column=1, padx=5, pady=5)

showinfo.mainloop()

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

def showing():

"""Автор: Царегородцева Елизавета

Цель: открытие окна для просмотра данных

Вход: event

Выход: -"""

try:

item = tree.selection()[0]

for i in range(len(table\_ind)):

if item == table\_ind[i]:

index = i

showinfo = tk.Toplevel()

showinfo.title("Просмотр информации")

showinfo.geometry('500x600+450+140')

showinfo.resizable(False, False)

title = tk.Label(showinfo, text="Название:")

city = tk.Label(showinfo, text="Город:")

region = tk.Label(showinfo, text="Регион:")

link = tk.Label(showinfo, text="Ссылка:")

tel = tk.Label(showinfo, text="Телефон:")

ad = tk.Label(showinfo, text="Адрес:")

rating = tk.Label(showinfo, text="Рейтинг:")

schedule = tk.Label(showinfo, text="Время работы:")

price = tk.Label(showinfo, text="Цена:")

info = tk.Label(showinfo, text="Описание:")

title.grid(row=0, column=0, sticky="w")

city.grid(row=1, column=0, sticky="w")

region.grid(row=2, column=0, sticky="w")

link.grid(row=3, column=0, sticky="w")

tel.grid(row=4, column=0, sticky="w")

ad.grid(row=5, column=0, sticky="w")

rating.grid(row=6, column=0, sticky="w")

schedule.grid(row=7, column=0, sticky="w")

price.grid(row=8, column=0, sticky="w")

info.grid(row=9, column=0, sticky="w")

title\_e = tk.Entry(showinfo, width=40)

city\_e = tk.Entry(showinfo, width=40)

region\_e = tk.Entry(showinfo, width=40)

link\_e = tk.Entry(showinfo, width=40)

tel\_e = tk.Entry(showinfo, width=40)

ad\_e = tk.Entry(showinfo, width=40)

rating\_e = tk.Entry(showinfo, width=40)

schedule\_e = tk.Entry(showinfo, width=40)

price\_e = tk.Entry(showinfo, width=40)

info\_e = tk.Text(showinfo, state='disabled', width=30,

height=20, wrap='word')

title\_e.grid(row=0, column=1, padx=5, pady=5)

city\_e.grid(row=1, column=1, padx=5, pady=5)

region\_e.grid(row=2, column=1, padx=5, pady=5)

link\_e.grid(row=3, column=1, padx=5, pady=5)

tel\_e.grid(row=4, column=1, padx=5, pady=5)

ad\_e.grid(row=5, column=1, padx=5, pady=5)

rating\_e.grid(row=6, column=1, padx=5, pady=5)

schedule\_e.grid(row=7, column=1, padx=5, pady=5)

price\_e.grid(row=8, column=1, padx=5, pady=5)

info\_e.grid(row=9, column=1, padx=5, pady=5)

title\_e.insert(0, landmarks[index][1])

city\_e.insert(0, landmarks[index][3])

region\_e.insert(0, landmarks[index][2])

link\_e.insert(0, landmarks[index][0])

tel\_e.insert(0, landmarks[index][5])

ad\_e.insert(0, landmarks[index][4])

rating\_e.insert(0, landmarks[index][7])

schedule\_e.insert(0, landmarks[index][6])

price\_e.insert(0, landmarks[index][8])

info\_e.configure(state='normal')

info\_e.insert('end', landmarks[index][9])

info\_e.configure(state='disabled')

title\_e['state'] = 'disabled'

city\_e['state'] = 'disabled'

region\_e['state'] = 'disabled'

rating\_e['state'] = 'disabled'

link\_e['state'] = 'disabled'

showinfo.mainloop()

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

def show\_all():

"""Автор: Царегородцева Елизавета

Цель: отображение всех элементов таблицы

Вход: event

Выход: -"""

for i in tree.get\_children():

tree.delete(i)

for i in range(len(landmarks)):

if count\_1[i] == 0:

tree.insert('', 'end', values=(landmarks[i][1],

landmarks[i][3],

landmarks[i][2],

landmarks[i][7],

landmarks[i][0]))

city\_group = tk.LabelFrame(tab\_1, text='Выбор параметров')

city\_group.grid(column=0, row=0, pady=10, padx=10)

label\_city = tk.Label(city\_group, text="Выберите город:")

label\_city.grid(column=0, row=1, pady=10, padx=10)

combobox\_1 = ttk.Combobox(city\_group, values=cities.tolist())

combobox\_1.grid(column=0, row=2, pady=10, padx=10)

combobox\_1.current(0)

button\_city = ttk.Button(city\_group, text="Выбрать", command=get\_city)

button\_city.grid(column=0, row=3, pady=10, padx=10)

button\_all = ttk.Button(city\_group, text="Показать все", command=show\_all)

button\_all.grid(column=0, row=4, pady=10, padx=10)

analysis\_group = tk.LabelFrame(tab\_1, text='Выбор метода анализа')

analysis\_group.grid(column=1, row=0, pady=10, padx=10)

combobox\_2 = ttk.Combobox(analysis\_group,

values=["Базовая статистика",

"Сводная таблица",

"Столбчатая диаграмма",

"Гистограмма",

"Диаграмма Бокса-Вискера",

"Диаграмма рассеивания"])

combobox\_2.grid(column=1, row=1, pady=10, padx=10)

combobox\_2.current(0)

button\_method = ttk.Button(analysis\_group, text="Выбрать",

command=get\_analysis)

button\_method.grid(column=1, row=2, pady=10, padx=10)

editing\_group = tk.LabelFrame(tab\_1, text='Редактирование таблицы')

editing\_group.grid(column=2, row=0, pady=10, padx=10)

button\_delete = ttk.Button(editing\_group, text="Удалить",

command=deleting)

button\_delete.grid(column=2, row=1, pady=10, padx=10)

button\_change = ttk.Button(editing\_group, text="Изменить значения",

command=changing)

button\_change.grid(column=2, row=2, pady=10, padx=10)

button\_new = ttk.Button(editing\_group, text="Добавить новую строку",

command=new\_item)

button\_new.grid(column=2, row=3, pady=10, padx=10)

more\_group = tk.LabelFrame(tab\_1, text='Дополнительно')

more\_group.grid(column=3, row=0, pady=10, padx=10)

button\_like = ttk.Button(more\_group, text="Добавить в понравившееся",

command=liked)

button\_like.grid(column=3, row=1, pady=10, padx=10)

button\_show = ttk.Button(more\_group, text="Просмотреть полностью",

command=showing)

button\_show.grid(column=3, row=2, pady=10, padx=10)

columns = ("Название", "Город", "Регион", "Рейтинг", "Ссылка")

tree = ttk.Treeview(tab\_1, columns=columns, height=20)

rate = []

for i in range(len(landmarks)):

rate.append(0)

rate[i] = landmarks[i][7]

table()

**second\_tab.py**

"""Строит вкладку 'Понравившееся'."""

import tkinter as tk

from tkinter import ttk, Tk

from tkinter import messagebox as mb

from tkinter import filedialog as fd

import pandas as pd

import numpy as np

from pandas import ExcelWriter as ew

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import base\_statistics

import out\_pivot\_table

import group\_diagram

def error():

"""Автор: Мотявин Матвей

Цель: выдать сообщение об ошибке

Вход: -

Выход: окно с сообщением об ошибке """

mb.showerror('Ошибка',

'Невозможно произвести анализ с данными параметрами.')

def secondtab(tab\_2, liked\_list, table\_ind, new\_ind, count\_2):

"""Автор: Царегородцева Елизавета

Цель: построение и заполнение вкладки «Понравившееся»

Вход: списки с данными и индексами об их наличии/отсутствии

Выход: -"""

def get\_analysis():

"""Автор: Мотявин Матвей

Цель: строит окна с выбором параметров для анализа

Вход: event

Выход: вызов подходящей функции"""

def base\_stat():

"""Автор: Царегородцева Елизавета

Цель: проведение анализа

Вход: список данных

Выход: список с результатами анализа"""

amount = 0

for i in range(len(count\_2)):

if count\_2[i] == 1:

amount += 1

if amount != 0:

if var.get() == 0:

error()

elif var.get() == 3:

error()

elif var.get() == 1:

output = []

count = 1

index = 0

for i in range(len(count\_2)-1, -1, -1):

if count\_2[i] == 1:

index = i

for i in range(index+1, len(liked\_list)):

if (liked\_list[index][3] == liked\_list[i][3] and count\_2[i] == 1):

count += 1

output.append({

'Значение': liked\_list[index][3],

'Частота': count,

'Процент от общего числа': str(count/amount\*100) + "%"

})

for i in range(index+1, len(liked\_list)):

repeat = 0

for rep in range(len(output)):

if (liked\_list[i][3] == output[rep]['Значение'] and count\_2[i] == 1):

repeat += 1

if repeat == 0:

count = 0

for j in range(index+1, len(liked\_list)):

if (liked\_list[i][3] == liked\_list[j][3] and count\_2[i] == 1):

count += 1

if count > 0:

output.append({

'Значение': liked\_list[i][3],

'Частота': count,

'Процент от общего числа': str(count/amount\*100) + "%"

})

elif var.get() == 2:

output = []

count = 1

index = 0

for i in range(len(count\_2)-1, -1, -1):

if count\_2[i] == 1:

index = i

for i in range(index+1, len(liked\_list)):

if (liked\_list[index][2] == liked\_list[i][2] and count\_2[i] == 1):

count += 1

output.append({

'Значение': liked\_list[index][2],

'Частота': count,

'Процент от общего числа': str(count/amount\*100) + "%",

})

for i in range(index+1, len(liked\_list)):

repeat = 0

for rep in range(len(output)):

if (liked\_list[i][2] == output[rep]['Значение'] and count\_2[i] == 1):

repeat += 1

if repeat == 0:

count = 0

for j in range(index+1, len(liked\_list)):

if (liked\_list[i][2] == liked\_list[j][2] and count\_2[i] == 1):

count += 1

if count > 0:

output.append({

'Значение': liked\_list[i][2],

'Частота': count,

'Процент от общего числа': str(count/amount\*100) + "%"

})

elif var.get() == 4:

dfcolumns = ['lm\_link', 'lm\_title', 'lm\_region',

'lm\_city', 'lm\_contacts\_ad',

'lm\_contacts\_tel', 'lm\_schedule',

'lm\_rating', 'lm\_price', 'lm\_info']

liked = []

j = int(0)

for i in range(len(liked\_list)):

liked.append(0)

for i in range(len(liked\_list)):

if count\_2[i] == 1:

liked[j] = dict(zip(dfcolumns, liked\_list[i]))

j += 1

df = pd.DataFrame.from\_dict(liked)

output = []

output.append({

'Переменная': 'Рейтинг',

'Максимум': df['lm\_rating'].max(),

'Минимум': df['lm\_rating'].min(),

'Среднее арифметическое': df['lm\_rating'].mean(),

'Выборочная дисперсия': df['lm\_rating'].std(0),

'Стандартное отклонение': df['lm\_rating'].var(ddof=1),

})

elif var.get() == 5:

dfcolumns = ['lm\_link', 'lm\_title', 'lm\_region',

'lm\_city', 'lm\_contacts\_ad',

'lm\_contacts\_tel', 'lm\_schedule',

'lm\_rating', 'lm\_price', 'lm\_info']

liked = []

j = int(0)

for i in range(len(liked\_list)):

liked.append(0)

for i in range(len(liked\_list)):

if count\_2[i] == 1:

liked[j] = dict(zip(dfcolumns, liked\_list[i]))

j += 1

df = pd.DataFrame.from\_dict(liked)

output = []

output.append({

'Переменная': 'Цена',

'Максимум': df['lm\_price'].max(),

'Минимум': df['lm\_price'].min(),

'Среднее арифметическое': df['lm\_price'].mean(),

'Выборочная дисперсия': df['lm\_price'].std(0),

'Стандартное отклонение': df['lm\_price'].var(ddof=1),

})

dialog1.destroy()

if (var.get() == 1 or var.get() == 2):

base\_statistics.base\_text(output)

if (var.get() == 4 or var.get() == 5):

base\_statistics.base\_num(output)

else:

mb.showerror('Ошибка', 'Список пуст.')

def sv\_table():

"""Автор: Царегородцева Елизавета

Цель: проведение анализа

Вход: список данных

Выход: обьект DataFrame"""

amount = 0

for i in range(len(count\_2)):

if count\_2[i] == 1:

amount += 1

if amount != 0:

if entry\_quality1.get() == 'Регион':

if entry\_quality2.get() == 'Город':

if agreg.get() == 'Цена':

new\_list = []

for i in range(len(count\_2)):

if count\_2[i] == 1:

new\_list.append({'Регион':

liked\_list[i][2],

'Город': liked\_list[i][3],

'Цена': liked\_list[i][8]})

df = pd.DataFrame.from\_dict(new\_list)

data = pd.pivot\_table(df, index=['Регион', 'Город'],

values=['Цена'],

aggfunc=np.sum)

else:

new\_list = []

for i in range(len(count\_2)):

if count\_2[i] == 1:

new\_list.append({'Регион': liked\_list[i][2],

'Город': liked\_list[i][3],

'Количество достопримечательностей':

int(1)})

df = pd.DataFrame.from\_dict(new\_list)

data = pd.pivot\_table(df, index=['Регион', 'Город'],

values=['Количество достопримечательностей'],

aggfunc=np.sum)

dialog2.destroy()

out\_pivot\_table.output(data)

dialog3.destroy()

else:

error()

else:

mb.showerror('Ошибка', 'Список пуст.')

def gr\_diagram():

"""Автор: Царегородцева Елизавета

Цель: анализ возможности построения столбчатой диаграммы

Вход: event

Выход: вызов функции или окно с ошибкой"""

amount = 0

for i in range(len(count\_2)):

if count\_2[i] == 1:

amount += 1

if amount != 0:

if entry\_quality2.get() == 'Тип достопримечательности':

if entry\_quality1.get() == 'Регион':

group\_diagram.output(2, liked\_list, count\_2)

elif entry\_quality1.get() == 'Город':

group\_diagram.output(3, liked\_list, count\_2)

else:

error()

else:

mb.showerror('Ошибка', 'Список пуст.')

choosen\_analysis = combobox\_2.get()

if choosen\_analysis == "Базовая статистика":

dialog1 = tk.Toplevel()

dialog1.title('Параметры базовой статистики')

dialog1.geometry('300x300+400+300')

dialog1.resizable(False, False)

var = tk.IntVar()

var.set(0)

title = ttk.Radiobutton(dialog1, text='Название', variable=var, value=0)

title.place(x=50, y=30)

city = ttk.Radiobutton(dialog1, text='Город', variable=var, value=1)

city.place(x=50, y=60)

region = ttk.Radiobutton(dialog1, text='Регион', variable=var, value=2)

region.place(x=50, y=90)

link = ttk.Radiobutton(dialog1, text='Ссылка', variable=var, value=3)

link.place(x=50, y=120)

rating = ttk.Radiobutton(dialog1, text='Рейтинг', variable=var, value=4)

rating.place(x=50, y=150)

price = ttk.Radiobutton(dialog1, text='Цена', variable=var, value=5)

price.place(x=50, y=180)

btn\_ok = tk.Button(dialog1, text='Начать анализ', command=base\_stat)

btn\_ok.place(x=50, y=250)

dialog1.mainloop()

elif choosen\_analysis == 'Сводная таблица':

dialog2 = tk.Toplevel()

dialog2.title('Параметры сводной таблицы')

dialog2.geometry('400x220+400+300')

dialog2.resizable(False, False)

label\_quality1 = tk.Label(dialog2, text='Качественный параметр 1')

label\_quality1.place(x=50, y=50)

label\_quality2 = tk.Label(dialog2, text='Качественный параметр 2')

label\_quality2.place(x=50, y=80)

label\_numerical = tk.Label(dialog2, text='Метод агрегации')

label\_numerical.place(x=50, y=110)

entry\_quality1 = ttk.Combobox(dialog2, values=['Название',

'Город',

'Регион',

'Телефон',

'Ссылка'])

entry\_quality1.current(0)

entry\_quality1.place(x=200, y=50)

entry\_quality2 = ttk.Combobox(dialog2, values=['Название',

'Город',

'Регион',

'Телефон',

'Ссылка'])

entry\_quality2.current(0)

entry\_quality2.place(x=200, y=80)

agreg = ttk.Combobox(dialog2, values=['Цена', 'Количество достопримечательностей'])

agreg.current(0)

agreg.place(x=200, y=110)

dialog2.btn\_ok = tk.Button(dialog2, text='Начать анализ', command=sv\_table)

dialog2.btn\_ok.place(x=200, y=170)

dialog2.mainloop()

elif choosen\_analysis == 'Столбчатая диаграмма':

dialog3 = tk.Toplevel()

dialog3.title('Параметры столбчатой диаграммы')

dialog3.geometry('400x220+400+300')

dialog3.resizable(False, False)

label\_quality1 = tk.Label(dialog3, text='Качественный столбец 1')

label\_quality1.place(x=50, y=50)

label\_quality2 = tk.Label(dialog3, text='Качественный столбец 2')

label\_quality2.place(x=50, y=100)

entry\_quality1 = ttk.Combobox(dialog3, values=['Название',

'Город', 'Регион', 'Телефон', 'Ссылка'])

entry\_quality1.current(0)

entry\_quality1.place(x=200, y=50)

entry\_quality2 = ttk.Combobox(dialog3, values=['Название',

'Город',

'Регион',

'Тип достопримечательности',

'Адрес'])

entry\_quality2.current(0)

entry\_quality2.place(x=200, y=100)

btn\_cancel = ttk.Button(dialog3, text='Закрыть',

command=dialog3.destroy)

btn\_cancel.place(x=300, y=170)

btn\_ok = ttk.Button(dialog3, text='Начать анализ',

command=gr\_diagram)

btn\_ok.place(x=200, y=170)

dialog3.mainloop()

elif choosen\_analysis == 'Гистограмма':

dialog4 = Tk()

dialog4.title('Параметры гистограммы')

dialog4.geometry('400x220+400+300')

dialog4.resizable(False, False)

label\_quality1 = tk.Label(dialog4, text='Качественный столбец')

label\_quality1.place(x=50, y=50)

label\_numerical = tk.Label(dialog4, text='Численный столбец')

label\_numerical.place(x=50, y=100)

dialog4.entry\_quality1 = ttk.Combobox(dialog4, values=[u'Название',

u'Город',

u'Регион'])

dialog4.entry\_quality1.current(0)

dialog4.entry\_quality1.place(x=200, y=50)

dialog4.entry\_numerical = ttk.Combobox(dialog4, values=[u'Цена',

u'Рейтинг'])

dialog4.entry\_numerical.current(0)

dialog4.entry\_numerical.place(x=200, y=100)

btn\_cancel = ttk.Button(dialog4, text='Закрыть',

command=dialog4.destroy)

btn\_cancel.place(x=300, y=170)

dialog4.btn\_ok = ttk.Button(dialog4, text='Начать анализ',

command=histogramm)

dialog4.btn\_ok.place(x=200, y=170)

dialog4.btn\_ok.bind('<Button-1>',

lambda event:

histogramm(dialog4.entry\_quality1.get(),

dialog4.entry\_numerical.get()))

dialog4.grab\_set() # перехват всех событий, происходящих в приложении

dialog4.focus\_set() # захват и удержание фокуса

dialog4.mainloop()

elif choosen\_analysis == 'Диаграмма Бокса-Вискера':

dialog5 = Tk()

dialog5.title('Параметры Диаграммы Бокса-Вискера')

dialog5.geometry('400x220+400+300')

dialog5.resizable(False, False)

label\_quality1 = tk.Label(dialog5, text='Качественный столбец')

label\_quality1.place(x=50, y=50)

label\_numerical = tk.Label(dialog5, text='Численный столбец')

label\_numerical.place(x=50, y=100)

dialog5.entry\_quality1 = ttk.Combobox(dialog5, values=[u'Город',

u'Регион'])

dialog5.entry\_quality1.current(0)

dialog5.entry\_quality1.place(x=200, y=50)

dialog5.entry\_numerical = ttk.Combobox(dialog5, values=[u'Цена',

u'Рейтинг'])

dialog5.entry\_numerical.current(0)

dialog5.entry\_numerical.place(x=200, y=100)

btn\_cancel = ttk.Button(dialog5, text='Закрыть',

command=dialog5.destroy)

btn\_cancel.place(x=300, y=170)

dialog5.btn\_ok = ttk.Button(dialog5, text='Начать анализ',

command=boxgraph)

dialog5.btn\_ok.place(x=200, y=170)

dialog5.btn\_ok.bind('<Button-1>',

lambda event:

boxgraph(dialog5.entry\_quality1.get(),

dialog5.entry\_numerical.get()))

dialog5.grab\_set() # перехват всех событий, происходящих в приложении

dialog5.focus\_set() # захват и удержание фокуса

dialog5.mainloop()

elif choosen\_analysis == 'Диаграмма рассеивания':

dialog6 = Tk()

dialog6.title('Параметры диаграммы рассеивания')

dialog6.geometry('400x220+400+300')

dialog6.resizable(False, False)

label\_quality1 = tk.Label(dialog6, text='Качественный столбец')

label\_quality1.place(x=50, y=50)

label\_numerical1 = tk.Label(dialog6, text='Численный столбец 1')

label\_numerical1.place(x=50, y=80)

label\_numerical2 = tk.Label(dialog6, text='Численный столбец 2')

label\_numerical2.place(x=50, y=110)

dialog6.entry\_quality1 = ttk.Combobox(dialog6, values=[u'Название',

u'Город',

u'Регион'])

dialog6.entry\_quality1.current(0)

dialog6.entry\_quality1.place(x=200, y=50)

dialog6.entry\_numerical1 = ttk.Combobox(dialog6, values=[u'Цена',

u'Рейтинг'])

dialog6.entry\_numerical1.current(0)

dialog6.entry\_numerical1.place(x=200, y=80)

dialog6.entry\_numerical2 = ttk.Combobox(dialog6, values=[u'Цена',

u'Рейтинг'])

dialog6.entry\_numerical2.current(0)

dialog6.entry\_numerical2.place(x=200, y=110)

btn\_cancel = ttk.Button(dialog6, text='Закрыть',

command=dialog6.destroy)

btn\_cancel.place(x=300, y=170)

dialog6.btn\_ok = ttk.Button(dialog6, text='Начать анализ',

command=scatter)

dialog6.btn\_ok.place(x=200, y=170)

dialog6.btn\_ok.bind('<Button-1>', lambda event:

scatter(dialog6.entry\_quality1.get(),

dialog6.entry\_numerical1.get(),

dialog6.entry\_numerical2.get()))

dialog6.grab\_set() # перехват всех событий, происходящих в приложении

dialog6.focus\_set() # захват и удержание фокуса

dialog6.mainloop()

def deleting():

"""Автор: Царегородцева Елизавета

Цель: удаление элемента из таблицы

Вход: event

Выход: измененные глобальные переменные"""

try:

item = tree.selection()[0]

tree.delete(item)

for i in range(len(new\_ind)):

if item == new\_ind[i]:

count\_2[i] -= 1

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

def sort\_column(tv, col, reverse):

"""Автор: Царегородцева Елизавета

Цель: сортировка качественных столбцов

Вход: таблица и столбец для сортировки

Выход: -"""

l = [(tv.set(k, col), k) for k in tv.get\_children('')]

l.sort(reverse=reverse)

for index, (val, k) in enumerate(l):

tv.move(k, '', index)

tv.heading(col, text=col, command=lambda \_col=col:

sort\_column(tv, \_col, not reverse))

def tree\_sortf(tv, col, reverse):

"""Автор: Царегородцева Елизавета

Цель: сортировка количественных столбцов

Вход: таблица и столбец для сортировки

Выход: -"""

l = [(tv.set(k, col), k) for k in tv.get\_children('')]

l.sort(key=lambda rate: float(rate[0]), reverse=reverse)

for index, (val, k) in enumerate(l):

tv.move(k, '', index)

tv.heading(col, command=lambda: tree\_sortf(tv, col, not reverse))

def creation():

"""Автор: Царегородцева Елизавета

Цель: заполнение шапки таблицы, размещение скроллбара

Вход: -

Выход: -"""

tree['show'] = 'headings'

tree.column("0", width=150, stretch=False, anchor='c')

tree.column("1", width=150, stretch=False, anchor='c')

tree.column("2", width=150, stretch=False, anchor='c')

tree.column("3", width=140, stretch=False, anchor='c')

tree.column("4", width=300, stretch=False, anchor='c')

tree.heading("4", text="Ссылка")

tree.place(x=30, y=220)

scrll = ttk.Scrollbar(tab\_2, orient="vertical",

command=tree.yview)

scrll.place(x=922, y=220, height=430)

tree.configure(yscrollcommand=scrll.set)

def inserting():

"""Автор: Царегородцева Елизавета

Цель: заполнение таблицы данными (вкладка «Главное окно»)

Вход: -

Выход: - """

if len(liked\_list) > 0:

for i in tree.get\_children():

tree.delete(i)

for i in range(len(liked\_list)):

tree.insert('', 'end', values=(liked\_list[i][1],

liked\_list[i][3],

liked\_list[i][2],

liked\_list[i][7],

liked\_list[i][0]))

for i in range(len(new\_ind)):

if count\_2[i] != 1:

tree.delete(new\_ind[i])

def saving():

"""Автор: Царегородцева Елизавета

Цель: сохранение новой таблицы

Вход: event

Выход: excel файл"""

if 1 not in count\_2:

mb.showerror('Ошибка', 'Вы ничего не добавили в список!')

else:

savefile = fd.asksaveasfilename(filetypes=(("Excel files", "\*.xlsx"),

("All files", "\*.\*")))

if savefile:

dfcolumns1 = ['lm\_title', 'lm\_region', 'lm\_city']

dfcolumns2 = ['lm\_schedule', 'lm\_rating', 'lm\_price', 'lm\_info']

primary = []

contacts = []

info = []

j = int(0)

for i in range(len(liked\_list)):

primary.append(0)

info.append(0)

for i in range(len(liked\_list)):

if count\_2[i] == 1:

primary[j] = dict(zip(dfcolumns1, liked\_list[i][1:4]))

contacts.append({'lm\_link': liked\_list[i][0],

'lm\_contacts\_ad': liked\_list[i][4],

'lm\_contacts\_tel': liked\_list[i][5]})

info[j] = dict(zip(dfcolumns2, liked\_list[i][6:10]))

j += 1

df1 = pd.DataFrame.from\_dict(primary)

df2 = pd.DataFrame.from\_dict(contacts)

df3 = pd.DataFrame.from\_dict(info)

wrt = ew(savefile + ".xlsx", engine='xlsxwriter')

df1.to\_excel(wrt, sheet\_name="Основная таблица")

df2.to\_excel(wrt, sheet\_name="Контакты")

df3.to\_excel(wrt, sheet\_name="Дополнительная информация")

wrt.save()

primary.clear()

contacts.clear()

info.clear()

def showing():

"""Автор: Царегородцева Елизавета

Цель: открытие окна для просмотра данных

Вход: event

Выход: -"""

try:

item = tree.selection()[0]

for i in range(len(new\_ind)):

if item == new\_ind[i]:

index = i

showinfo = tk.Toplevel()

showinfo.title("Просмотр информации")

showinfo.geometry('500x600+450+140')

showinfo.resizable(False, False)

title = tk.Label(showinfo, text="Название:")

city = tk.Label(showinfo, text="Город:")

region = tk.Label(showinfo, text="Регион:")

link = tk.Label(showinfo, text="Ссылка:")

tel = tk.Label(showinfo, text="Телефон:")

ad = tk.Label(showinfo, text="Адрес:")

rating = tk.Label(showinfo, text="Рейтинг:")

schedule = tk.Label(showinfo, text="Время работы:")

price = tk.Label(showinfo, text="Цена:")

info = tk.Label(showinfo, text="Описание:")

title.grid(row=0, column=0, sticky="w")

city.grid(row=1, column=0, sticky="w")

region.grid(row=2, column=0, sticky="w")

link.grid(row=3, column=0, sticky="w")

tel.grid(row=4, column=0, sticky="w")

ad.grid(row=5, column=0, sticky="w")

rating.grid(row=6, column=0, sticky="w")

schedule.grid(row=7, column=0, sticky="w")

price.grid(row=8, column=0, sticky="w")

info.grid(row=9, column=0, sticky="w")

title\_e = tk.Entry(showinfo, width=40)

city\_e = tk.Entry(showinfo, width=40)

region\_e = tk.Entry(showinfo, width=40)

link\_e = tk.Entry(showinfo, width=40)

tel\_e = tk.Entry(showinfo, width=40)

ad\_e = tk.Entry(showinfo, width=40)

rating\_e = tk.Entry(showinfo, width=40)

schedule\_e = tk.Entry(showinfo, width=40)

price\_e = tk.Entry(showinfo, width=40)

info\_e = tk.Text(showinfo, state='disabled', width=30,

height=20, wrap='word')

title\_e.grid(row=0, column=1, padx=5, pady=5)

city\_e.grid(row=1, column=1, padx=5, pady=5)

region\_e.grid(row=2, column=1, padx=5, pady=5)

link\_e.grid(row=3, column=1, padx=5, pady=5)

tel\_e.grid(row=4, column=1, padx=5, pady=5)

ad\_e.grid(row=5, column=1, padx=5, pady=5)

rating\_e.grid(row=6, column=1, padx=5, pady=5)

schedule\_e.grid(row=7, column=1, padx=5, pady=5)

price\_e.grid(row=8, column=1, padx=5, pady=5)

info\_e.grid(row=9, column=1, padx=5, pady=5)

title\_e.insert(0, liked\_list[index][1])

city\_e.insert(0, liked\_list[index][3])

region\_e.insert(0, liked\_list[index][2])

link\_e.insert(0, liked\_list[index][0])

tel\_e.insert(0, liked\_list[index][5])

ad\_e.insert(0, liked\_list[index][4])

rating\_e.insert(0, liked\_list[index][7])

schedule\_e.insert(0, liked\_list[index][6])

price\_e.insert(0, liked\_list[index][8])

title\_e['state'] = 'disabled'

city\_e['state'] = 'disabled'

region\_e['state'] = 'disabled'

rating\_e['state'] = 'disabled'

info\_e.configure(state='normal')

info\_e.insert('end', liked\_list[index][9])

info\_e.configure(state='disabled')

link\_e['state'] = 'disabled'

showinfo.mainloop()

except:

mb.showerror('Ошибка', 'Вы ничего не выбрали!')

analysis\_group = tk.LabelFrame(tab\_2, text='Выбор метода анализа')

analysis\_group.grid(column=0, row=0, pady=10, padx=10)

combobox\_2 = ttk.Combobox(analysis\_group,

values=["Базовая статистика",

"Сводная таблица",

"Столбчатая диаграмма",

"Гистограмма",

"Диаграмма Бокса-Вискера",

"Диаграмма рассеивания"])

combobox\_2.grid(column=0, row=1, pady=10, padx=10)

combobox\_2.current(0)

button\_method = ttk.Button(analysis\_group, text="Выбрать",

command=get\_analysis)

button\_method.grid(column=0, row=2, pady=10, padx=10)

editing\_group = tk.LabelFrame(tab\_2, text='Редактирование таблицы')

editing\_group.grid(column=1, row=0, pady=10, padx=10)

button\_like = ttk.Button(editing\_group, text="Удалить из понравившегося",

command=deleting)

button\_like.grid(column=1, row=1, pady=10, padx=10)

save\_group = tk.LabelFrame(tab\_2, text='Экспорт данных')

save\_group.grid(column=2, row=0, pady=10, padx=10)

button\_save = ttk.Button(save\_group, text="Сохранить изменения",

command=saving)

button\_save.grid(column=2, row=0, pady=10, padx=10)

more\_group = tk.LabelFrame(tab\_2, text='Дополнительно')

more\_group.grid(column=3, row=0, pady=10, padx=10)

button\_show = ttk.Button(more\_group, text="Просмотреть полностью",

command=showing)

button\_show.grid(column=3, row=1, pady=10, padx=10)

columns = ("Название", "Город", "Регион", "Рейтинг", "Ссылка")

tree = ttk.Treeview(tab\_2, columns=columns, height=20)

columns\_sort = ("Название", "Город", "Регион")

column = ("Рейтинг",)

rate = []

for i in range(len(liked\_list)):

rate.append(0)

rate[i] = liked\_list[i][7]

for col in columns\_sort:

tree.heading(col, text=col, command=lambda \_col=col:

sort\_column(tree, \_col, False))

for col in column:

tree.heading(col, text=col, command=lambda:

tree\_sortf(tree, col, False))

creation()

inserting()

def histogramm(chosen1, chosen2):

"""Автор: Мотявин Матвей

Цель: построение гистограммы

Вход: названия столбцов

Выход: -"""

def save\_plot():

"""Автор: Мотявин Матвей

Цель: сохранение графического отчета

Вход: название фигуры

Выход: файл png"""

name = fd.asksaveasfilename(filetypes=(("PNG", "\*.png"), ("all files", "\*.\*")))

fmt = name + ".png"

figure1.savefig(fmt)

df = pd.DataFrame(liked\_list[:][:])

if chosen1 == 'Название':

L1Values = df.iloc[:, 1]

elif chosen1 == 'Город':

L1Values = df.iloc[:, 3]

elif chosen1 == 'Регион':

L1Values = df.iloc[:, 2]

if chosen2 == 'Цена':

L2Values = df.iloc[:, 8]

if chosen2 == 'Рейтинг':

L2Values = df.iloc[:, 7]

predict = tk.Tk()

predict.title('Гистограмма')

canvas1 = tk.Canvas(predict, width=600, height=0)

canvas1.pack()

figure1 = plt.Figure(figsize=(9, 8), dpi=90)

ax = figure1.add\_subplot(111)

ax.bar(L1Values, L2Values, color='orange')

L1Values = ax.set\_xticklabels(L1Values,

fontsize=10,

rotation=25,

verticalalignment='center')

hist1 = FigureCanvasTkAgg(figure1, predict)

hist1.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH)

ax.legend()

ax.set\_title(f"{chosen1} Vs. {chosen2}")

SaveBut = ttk.Button(predict, text="Сохранить", width=20,

command=save\_plot)

SaveBut.place(y=20)

predict.grab\_set()

predict.focus\_set()

predict.mainloop()

def boxgraph(chosen1, chosen2):

"""Автор: Мотявин Матвей

Цель: построение диаграммы Бокса-Вискера

Вход: названия столбцов

Выход: - """

def check(list):

"""Автор: Мотявин Матвей

Цель: проверить все ли данные в выбранном стоблце идентичны

Вход: список

Выход: список"""

return all(i == list[0] for i in list)

def save\_plot():

"""Автор: Мотявин Матвей

Цель: сохранение графического отчета

Вход: название фигуры

Выход: файл png"""

name = fd.asksaveasfilename(filetypes=(("PNG", "\*.png"),

("all files", "\*.\*")))

fmt = name+".png"

figure1.savefig(fmt)

df = pd.DataFrame(liked\_list[:][:])

if chosen1 == 'Город':

L1Values = df.iloc[:, 3]

elif chosen1 == 'Регион':

L1Values = df.iloc[:, 2]

if check(L1Values) == True:

if chosen2 == 'Цена':

L2Values = df.iloc[:, 8]

elif chosen2 == 'Рейтинг':

L2Values = df.iloc[:, 7]

predict = tk.Tk()

predict.title('Диаграмма Бокса-Вискера')

canvas1 = tk.Canvas(predict, width=600, height=0)

canvas1.pack()

figure1 = plt.Figure(figsize=(8, 8), dpi=100)

ax = figure1.add\_subplot(111)

ax.boxplot(L2Values,

patch\_artist=True,

medianprops={'color': "#297083"},

boxprops={'color': "#539caf", 'facecolor': "#539caf"},

whiskerprops={'color': "#539caf"},

capprops={'color': "#539caf"})

ax.set\_xticklabels(L1Values)

hist1 = FigureCanvasTkAgg(figure1, predict)

hist1.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH)

ax.legend()

ax.set\_xlabel(chosen1)

ax.set\_ylabel('ось ординат (YAxis)')

ax.set\_title('Layer1 Temperature Vs. Soil Temperature')

SaveBut = ttk.Button(predict, text="Сохранить",

width=20, command=save\_plot)

SaveBut.place(y=20)

predict.grab\_set()

predict.focus\_set()

predict.mainloop()

else:

mb.showerror('Ошибка',

'Невозможно произвести анализ с данными параметрами. Проверьте чтобы все значения в выбранном вами качественном столбце были одинаковыми!')

def scatter(chosen1, chosen2, chosen3):

"""Автор: Мотявин Матвей

Цель: построение диаграммы рассеивания

Вход: названия столбцов

Выход: -"""

def save\_plot():

"""Автор: Мотявин Матвей

Цель: сохранение графического отчета

Вход: название фигуры

Выход: файл png"""

name = fd.asksaveasfilename(filetypes=(("PNG", "\*.png"),

("all files", "\*.\*")))

fmt = name+".png"

figure1.savefig(fmt)

df = pd.DataFrame(liked\_list[:][:])

if chosen1 == 'Название':

L1Values = df.iloc[:, 1]

elif chosen1 == 'Город':

L1Values = df.iloc[:, 3]

elif chosen1 == 'Регион':

L1Values = df.iloc[:, 2]

if chosen2 == 'Цена':

L2Values = df.iloc[:, 8]

elif chosen2 == 'Рейтинг':

L2Values = df.iloc[:, 7]

if chosen3 == 'Цена':

L3Values = df.iloc[:, 8]

elif chosen3 == 'Рейтинг':

L3Values = df.iloc[:, 7]

data = {'Name': L1Values,

'Name1': L2Values,

'Name2': L3Values}

midwest = pd.DataFrame(data)

categories = np.unique(midwest['Name'])

colors = [plt.cm.tab10(i/float(len(categories)-1)) for i in range(len(categories))]

predict = tk.Tk()

predict.title('Диаграмма рассеивания')

canvas1 = tk.Canvas(predict, width=600, height=0)

canvas1.pack()

figure1 = plt.Figure(figsize=(16, 10), dpi=80, facecolor='w',

edgecolor='k')

ax = figure1.add\_subplot(111)

for i, Name in enumerate(categories):

ax.scatter('Name1', 'Name2',

data=midwest.loc[midwest.Name == Name, : ],

s=20, c=colors[i], label=str(Name))

hist1 = FigureCanvasTkAgg(figure1, predict)

hist1.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH)

ax.legend()

ax.set\_xlabel(chosen2)

ax.set\_ylabel(chosen3)

ax.set\_title(f"Диаграмма рассеивания")

SaveBut = ttk.Button(predict, text="Сохранить", width=20,

command=save\_plot)

SaveBut.place(y=20)

predict.grab\_set()

predict.focus\_set()

predict.mainloop()